UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE COMPUTAÇÃO

GRUPO DELTA
ÁTILA DA ROCHA COSTA E SILVA
BÉUREN FELIPE BECHLIN
FELIPE BERTOLDO COLOMBO DE SOUZA

Implementação do jogo War sando TypeScript

Relatório apresentado como requisito parcial para a obtenção de conceito na Disciplina de Modelos de Linguagens de Programação

Prof. Dr. Lucas Mello Schnorr
Orientador

Porto Alegre
2018

SUMÁRIO

1 INTRODUÇÃO

O trabalho visa a solução de um problema através de diferentes paradigmas de programação: orientação a objetos e funcional.

Com as duas implementações, teremos visto na prática as vantagens e desvantagens de cada paradigma na abordagem ao problema, tendo condições de estabelecer um comparativo entre diferentes pontos.

O problema escolhido foi o jogo War, versão brasileira do norte americano Risk, que será abordado a seguir.

1.1 Visão geral da linguagem

Dentro das linguagens disponibilizadas, TypeScript é a que possui maior suporte para elementos de interfaces e bibliotecas em geral, já que é um superset de JavaScript que, através do TypeScript Transpiler, é transformada para JavaScript. Outro fator determinante é a necessidade de solucionar o problema também de uma forma funcional, e JavaScript é uma linguagem essencialmente funcional com vasta utilização de closures e padrões de projeto, como módulo.

Mesmo TypeScript sendo implementada para restringir o uso de JavaScript, além de adicionar tipagem estática e outros elementos de OOP, é possível utilizar esses padrões herdados da linguagem JavaScript, ajudando bastante em uma solução funcional. Como já citado, o TypeScript foi desenvolvido com o intuito de criar uma linguagem mais estável para aplicações web e com melhor manutenibilidade que o JS. Adicionando elementos de checagem estática de tipos, diferentemente do JS, orientação a objetos com maior poder que as especificações de EcmaScript 6.

Dessa maneira, possuímos uma linguagem completa para solucionar um problema com grande interação com usuário e elementos gráficos como é um caso de um jogo. Para exemplificar, é possível utilizar o framework de single page applications criado pelo Google chamado Angular, usado por grandes empresas. Também é possível utilizar uma biblioteca interface com usuário chamada React, desenvolvida e mantida pelo Facebook usada por Netflix, AirBnb e outros.

Para auxílio da programação funcional existe a implementação da biblioteca underscore para TypeScript. Além disso a linguagem possui um ótimo ambiente de desenvolvimento com vários utilitários desenvolvidos por terceiros para gerenciamento de

pacotes e dependências, gerenciadores de tarefas e frameworks para testes.

1.2 Descrição do problema

Em TypeScript será implementado o jogo War, versão brasileira do norteamericano Risk.

No jogo, o participante controla um exército que recebe um objetivo que deverá ser cumprido. O objetivo é conquistar determinado território. Se, ao final de uma batalha, o participante destruir todos os exércitos de defesa do adversário, ele terá conquistado o território. Quando um objetivo é declarado como concluído, o participante que o declarou conquista o território de combate e recebe novos exércitos, aumentando seu poder.

Em cada rodada, o participante poderá receber novos exércitos, caso conquiste territórios, usá-los conforme sua estratégia, atacar outros exércitos ou mover exércitos.

Vence o participante que atingir o objetivo recebido no início do jogo.

2 DESENVOLVIMENTO

2.1 Dependências

A única dependência que deve estar instalada no host para o processo de build do projeto é o **NodeJS**. O NodeJS é uma versão da V8, engine de JavaScript desenvolvida pelo Google para pré-compilação e execução de JavaScript em seu navegador Google Chrome, portada para plataforma.

Um dos pontos importantes dessa engine de JavaScript é que ela é totalmente single thread, mas usa um modelo de eventos e I/O não bloqueante para lidar com essa limitação. Esse modelo tem se mostrado uma alternativa aos tradicionais com multi threads, tanto que o NodeJS vem sendo vastamente usado como servidor aplicação para aplicações web.

Juntamente com o NodeJS, é instalado o gerenciador de pacotes **NPM**, que possui o maior ecosistema de bibliotecas open source do mundo. Essas dependências são instaladas localmente dentro de uma pasta na mesma raiz do projeto, independente da plataforma e sem nenhuma ligação com sistema operacional do usuário. Isso aumenta a portabilidade do projeto e automatiza o processo de construção da aplicação, já que com um comando é baixado e instalado todas as dependências listadas no projeto.

2.1.1 Instalando a aplicação

Como citado anteriormente o projeto usa o **NodeJS** e **NPM**. No primeiro momento é necessário instalar as dependências externas do projeto, como até mesmo o **Typescript**, **Babel transpiler**, **Redux**, **React** e outros.

Para realizar a instalação, execute via terminal o seguinte comando no diretório raíz do projeto:

```
npm install
```

Esse comando chama o gerenciador de pacotes **NPM** solicitando a instalação dos pacotes que estão listados no arquivo `PACKAGE.json`. Após instaladas as dependências, é necessário "compilar"o projeto, nas próximas sessões será descrito como ocorre esse processo.

Para iniciar o ambiente de desenvolvimento é necessário executar o seguinte co-

mando:

```
npm run start
```

Esse comando chama o script start também descrito no arquivo `PACKAGE.json`. Como resultado, ele compila o projeto, abre o navegador padrão do host (apontando para o localhost porta 3000), e já sobe um servidor de aplicação node que serve o conteúdo estático ligado com todas as interfaces de rede do Host na porta 3000. Além disso, é adicionado um watcher na pasta, com Inotify em linux distros, para recompilar o projeto em casos de modificações e automaticamente recarregar a página do navegador.

2.2 TypeScript vs. JavaScript

Ao pé da letra JavaScript não é uma linguagem, mas sim uma definição feita pela ECMA foundation para servir como um contrato entre os navegadores e desenvolvedores de aplicações web sobre o que está disponível na sua engine de script. Isso nem sempre é totalmente cumprido e, com o processo de evolução da linguagem, vários navagores não implementam a especificação por completo.

Esses problemas acabaram levando a soluções alternativas pela comunidade, como implementar as funcionalidadesnovas da linguagem usando elementos disponíveis no momento. Essa metodologia ficou conhecida como polyfill. Isso também foi evoluindo, criando um conjunto grandes de polyfills e então se começou a realizar o processo de transpilação, onde o código fonte TypeScript é transformado novamente em JavaScript.

2.3 Paradigma funcional

Cada uma das entidades do projeto - país, jogo, menu, jogador - é composto de três principais partes: as ações, os redutores e os tipos.

Os arquivos de **constantes** (types.ts) definem os tipos que cada uma das ações das entidades pertence, bem como outras constantes utilizadas em outras partes do projeto. Para fins de melhor legibilidade e modularização, cada entidade do projeto possui um arquivo de constantes a ela associado.

As **ações** - actions - de cada entidade são funções que correspondem a comandos específicos que elas podem executar de modo a gerar informações que possibilitem a posterior alteração do estado atual do jogo (feita pelos redutores). Cada ação possui um

tipo associado, definido no arquivo de tipos citado acima, e carrega consigo algum dado que é levado à store do projeto.

Os **redutores** são os responsáveis pelo controle do estado atual do jogo. Eles recebem a informação gerada por uma ação e realizam a criação de um novo estado de jogo de acordo com o novo dado. Ou seja, os reducers têm em mãos o estado atual do jogo, uma ação disparada por determinada entidade, e então produzem e retornam um novo estado composto pelas informações do estado antigo atualizadas de acordo com o dado trazido pela ação.

As entidades, com seus redutores, ações e constantes, constituem a **store** da aplicação. Uma store é um objeto que armazena o estado atual da aplicação, provedora de uma interface que faz a mediação do acesso ao estado.

2.4 Paradigma de orientação a objetos

3 RESULTADOS

4 CONCLUSÃO

REFERÊNCIAS

The TypeScript Reference,
`https://www.typescriptlang.org/docs/home.html`

The ReactJS Reference,
`https://reactjs.org/docs/react-api.html`

Using Redux with React,
`https://redux.js.org/basics/usage-with-react`

Underscore JS,
`http://underscorejs.org/`

React and Redux TypeScript Guide,
`https://github.com/piotrwitek/react-redux-typescript-guide`